Popular Computing

September 1976 Volume 4 Number 9

00	01	02	03	04	05	06	07	08	09
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34 20	35	36 19	37	38	39
40	41	42	43	44 14	45 10	46 4	47 3	48 21	49 22
50	51	52	53	54 17	55 11	56 6	57 1	58	59
60	61	62	63	64	65	66 5	67 2	68 8	69
70	71	72	73	74	75 9	76 7	77 12	78	79
80	81	82	83	84	85 13	86	87 16	88	89
90	91	92	93	94 18	95 15	96	97	98	99

A Constrained Random Walk

In the 10 x 10 array on the cover, 22 of the cells have been selected by the following scheme. Two-digit random numbers were drawn (the numbers used for this example were taken from the table of random digits given on page 13 of PC33, two digits at a time, reading down the page). The first number was 57, which indicated the first cell of the cover pattern. Successive random 2-digit numbers point to other cells, but a "hit" can only be scored in a cell adjacent to one previously hit. Thus, at the start, only cells 47, 56, 58, and 67 are eligible, and the second hit is scored when one of these (namely, 67) appears in the random stream (which was the 14th random draw from the table being used). At that point, the number of eligible cells enlarges to include:

47, 56, 58, 66, 68, and 77.

This procedure continues until all the cells are hit; the score is then the number of random numbers used. For a 10 x 10 array, this is likely to take over 900 numbers. To fill the 22 cells shown on the cover, it took 150 random selections of 2-digit numbers.

For other size arrays, each game would take a different number of draws of the random numbers, and the result, for any given size of array, is a distribution. The accompanying table shows the results for 100 games played with arrays of size 2, 3, 4,...,20; the calculations were made by Associate Editor David Babcock.

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 to the above rates. For all other countries, add $3.50 per year to the above rates. Back issues $2.50 each.

The table shows that there is a wide range between the smallest number of draws needed to fill an array and the largest number. The average number of draws plots as a fairly smooth curve. When plotted on log/log paper, the points appear to lie on a straight line.

What is the functional relationship between the size of the array and the number of draws of random numbers needed to fill that array? What values should be used to determine that function (e.g., are the standard deviations of any use)? How does one determine the type of function to be fitted?

More data is needed, particularly for arrays larger than 20 x 20.

PROBLEM **137**

Size	Minimum	Maximum	Mean	Standard deviation
2	4	28	9.27	4.2897
3	11	65	30.43	9.8916
4	38	132	71.15	18.4803
5	69	249	132.73	32.7278
6	120	397	231.41	54.3751
7	230	622	355.39	67.4347
8	334	874	517.42	105.3863
9	476	1201	718.00	142.8735
10	550	1405	906.27	159.4093
11	797	1967	1192.07	208.4468
12	937	2230	1519.04	263.4189
13	1149	2885	1834.08	305.5707
14	1512	3306	2287.07	340.8515
15	1843	4221	2719.07	435.7215
16	2172	4595	3276.58	528.3780
17	2722	5206	3867.54	514.2681
18	3229	6724	4498.46	683.6533
19	3727	7536	5255.13	797.8735
20	4181	8697	6210.23	953.5309

For example, on a 10 x 10 array, all cells were hit in as few as 550 tries and at most 1405 tries; the average number of tries was 906 with a standard deviation of 159. □

In our Contest No. 8 (PC39-4), each entry consists of nine 3-digit positive numbers in the range 100-999. To check the entries, a Fortran program is to be written to accept the nine numbers for one entry, punched on a card in columns 1-27. The program should reject, with an error message, any card for which the data numbers are not in the required range.

The output from the program should show the input data and, on the same print line, the sum, the sum of squares, the mean (correct to 3 decimal places), and the sample standard deviation (correct to 3 decimal places). The End-of-file procedure is simply running out of input data cards. The following test data should produce the indicated results:

									Sum	Sum of squares	Mean	Standard deviation
100	101	102	103	104	105	106	107	108	936	97404	104.000	2.739
100	200	300	400	500	600	700	800	900	4500	2850000	500.000	273.861
500	501	502	099	504	505	506	507	508	ØUT ØF RANGE			
500	501	502	503	504	505	506	507	508	4536	2286204	504.000	2.739
123	234	345	456	567	678	789	890	901	4983	3404441	553.667	284.060
999	999	999	999	999	100	100	100	100	5395	5030005	599.444	473.815
123	123	123	123	123	123	123	123	999	1983	1119033	220.333	292.000
123	456	789	987	654	321	234	345	456	4365	2732229	485.000	277.309
991	992	993	994	995	996	997	998	999	8955	8910285	995.000	2.739

All of the above is straightforward and is readily coded in Fortran. The Contest is this: using only ANSI Fortran, the $25 prize goes to the program using the least number of characters in the program, counting blanks. The number of characters per card will be counted up to the last card column used by the compiler. Thus, the Fortran statement:

```
N=N+1
```

would count as eleven characters, since the statement must begin in column 7 of the card. Similarly, the statement:

```
45 IF(N.NE.NSAVE.ØR.K.NE.KSAVE)GØTØ10
```

would count as 40 characters.

The characters used on COMMENTS cards will not be counted. Each entry must include a listing of the program, the test results, and the count of the characters. All entries must be received by POPULAR COMPUTING, Box 272, Calabasas, California 91302 by November 30, 1976.

"Sequences"

In the May issue of 65 Notes, a program was given by Dr. Mordecai Schwartz to generate "Sequences." As implemented on an SR-52 calculator, the program will produce an endless string of sequences, of the sort found on intelligence test ("What are the next two terms of this sequence?"). Here are some examples of the output of Dr. Schwartz's ingenious program:

8 22 57 155 400 1086 2801 7603 19608 ...

-4 2 -6 0 -10 -6 --20 -22 -46 -64 ...

25 149 895 5369 32215 193289 1159735 ...

4 1 7 6 15 19 36 53 91 142 235 ...

18 42 108 252 648 1512 3888 9072 ...

2 4 -8 -4 12 12 -28 -20 52 44 -108 ...

5 28 163 950 5537 32272 188095 1096298 ...

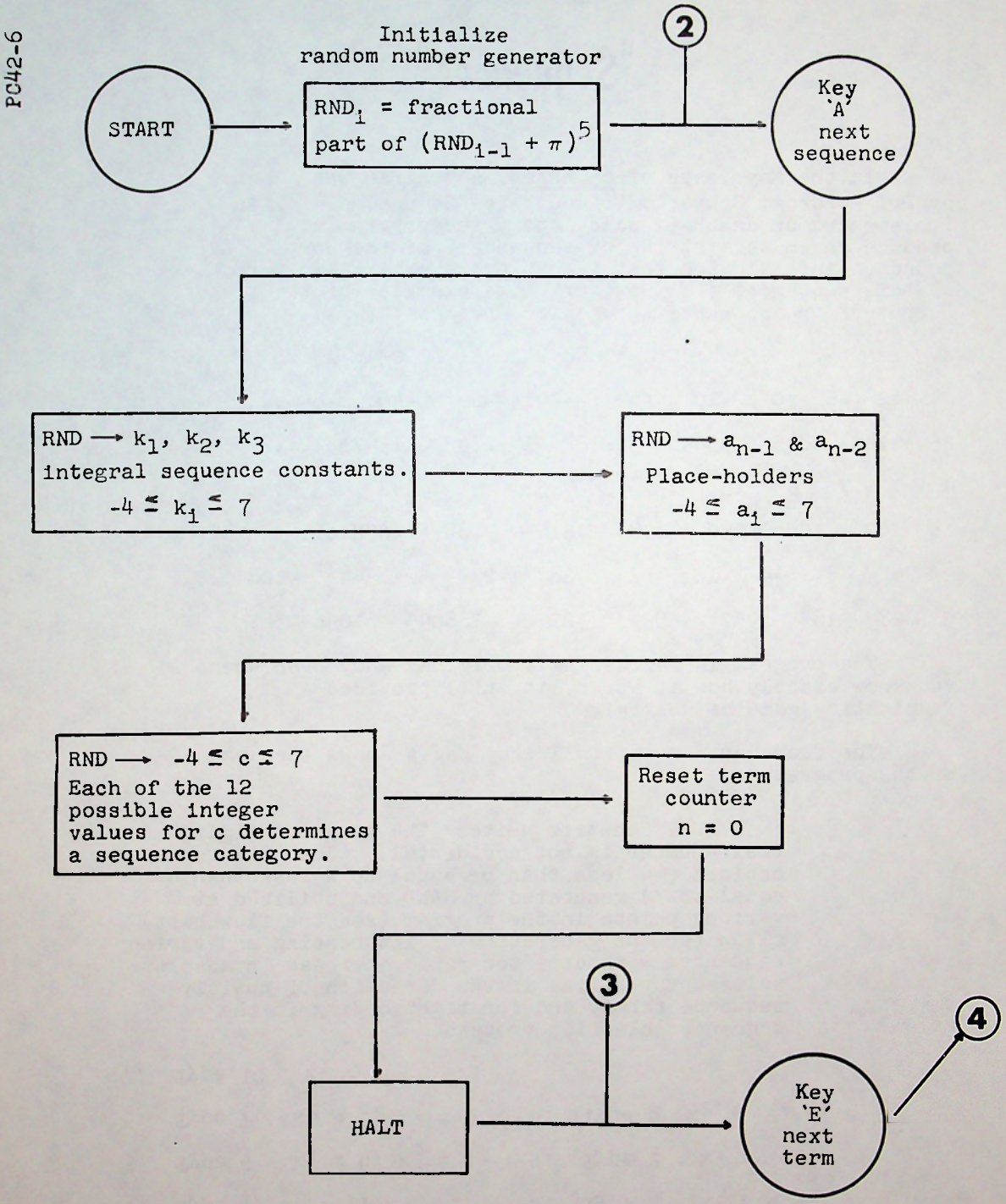
The program is refreshingly original and, even when you know exactly how it works, it still provides a fascinating game of solitaire.

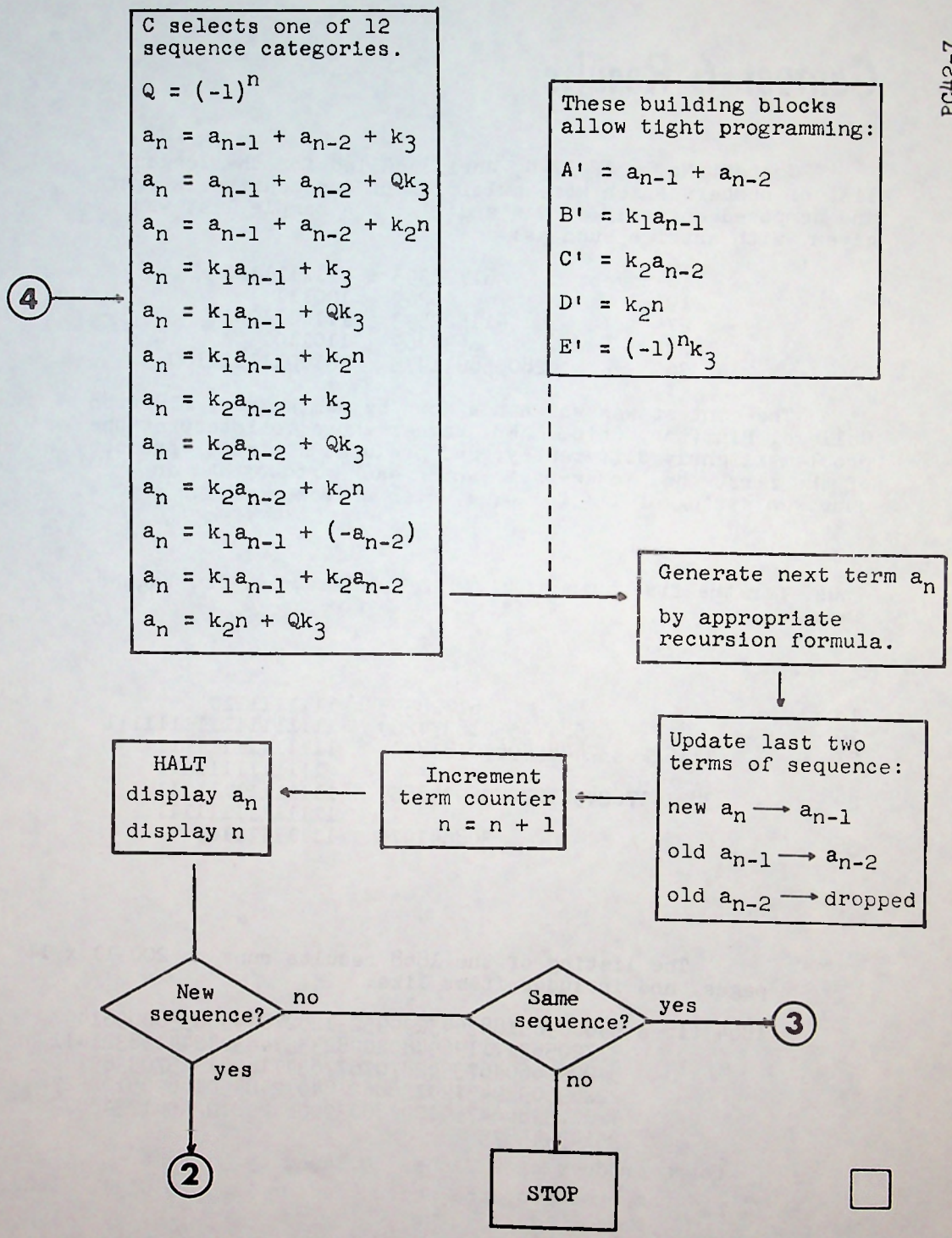
The flowchart on the following pages shows the logic of the program.

Dr. Schwartz writes "The range of digits and their number is not accidental. The 12 integer-choices (-4 less than or equal to k less than or equal to 7) generated by RAND and utilized at various points in the program (see the flowchart) allow for the generation of interesting and varied sequences without a too-rapid increase in absolute values, for a bias in the direction of positive sequence terms, and for tight coding in the sequence selection process."

For the 7 sequences given above, the patterns are given here. The types referred to are those given at Reference 4 of the flowchart, numbered from 1 to 12.

1. Type 7 $k_2 = 7$, $k_3 = 1$ 2. Type 2 $k_3 = 4$
3. Type 4 $k_1 = 6$, $k_3 = 1$ 4. Type 2 $k_3 = 2$
5. Type 7 $k_2 = 6$, $k_3 = 0$ 6. Type 8 $k_2 = -2$, $k_3 = 4$
7. Type 10 $k_1 = 6$


START
Initialize
random number generator
RND$_i$ = fractional part of $(RND_{i-1} + \pi)^5$
2
Key 'A' next sequence
RND → k_1, k_2, k_3
integral sequence constants.
$-4 \leq k_i \leq 7$
RND → a_{n-1} & a_{n-2}
Place-holders
$-4 \leq a_i \leq 7$
RND → $-4 \leq c \leq 7$
Each of the 12 possible integer values for c determines a sequence category.
Reset term counter
n = 0
3
HALT
Key 'E' next term
4


C selects one of 12 sequence categories.
Q = (-1)^n
a_n = a_n-1 + a_n-2 + k_3
a_n = a_n-1 + a_n-2 + Qk_3
a_n = a_n-1 + a_n-2 + k_2n
a_n = k_1a_n-1 + k_3
a_n = k_1a_n-1 + Qk_3
a_n = k_1a_n-1 + k_2n
a_n = k_2a_n-2 + k_3
a_n = k_2a_n-2 + Qk_3
a_n = k_2a_n-2 + k_2n
a_n = k_1a_n-1 + (-a_n-2)
a_n = k_1a_n-1 + k_2a_n-2
a_n = k_2n + Qk_3
4
These building blocks allow tight programming:
A' = a_n-1 + a_n-2
B' = k_1a_n-1
C' = k_2a_n-2
D' = k_2n
E' = (-1)^nk_3
Generate next term a_n by appropriate recursion formula.
Update last two terms of sequence:
new a_n → a_n-1
old a_n-1 → a_n-2
old a_n-2 → dropped
Increment term counter n = n + 1
HALT
display a_n
display n
New sequence?
no
yes
Same sequence?
yes
3
no
2
STOP

Contest 6 Results

Contest No. 6 (PC37-4, April) called for the longest list of numbers which were multiples of consecutive integers and composed entirely of 0's and 1's. A sample list was given, with entries such as:

18	times	61728395 =	1111111110
19		5269	100111
27		411522593	11111110011
29		37969	1101101
36		280558641975	10100111111100

The contest was won hands down by Sam Wagner, Bluffton College, Bluffton, Ohio. Mr. Wagner chose to interpret the problem slightly differently, and produced a list of products of the first 1868 integers in which each product had an unbroken string of 1's followed by an unbroken string of 0's.

Thus, for the five numbers given in our example, Mr. Wagner shows:

18	times	61728395 =	1111111110
19		5847953216374269	1111111111111111111
27		411522633744855967078189 3	11111111111111
			11111111111111
29		38314176245210727969348659	11111111111111
			11111111111111
36		308641975	11111111100

The listing of the 1868 results runs to 200 11 x 14 pages, and includes items like:

1864 times 59608965188364329995231282784930853600381497
3772055317119694802098235574630424415832141 1
5402956604673342870767763471626132570338578 9
2226990939437291368621840724845016690510252 7
4201239866475917978063900810681926561754887 9
35145445875

for a product of 232 ones followed by 3 zeros.

In many cases, the second factor is 1820 digits long.

Comments on the article "ASSEMBLERS" in issue 41, page 8
--by Irwin Greenwald

At the risk of being offensive, let me begin by saying that the whole article strikes me as pomposity borne out of ignorance. Sorry, but you're simply out of date on the subject. Let me introduce you to the state of the art as of 1976.

First, you are confused between the tasks of an assembler, the operating system it works under, the link editor, and the loader. Then, you do not seem to appreciate the trade-offs that the writer of an assembler must consider. For example, the choice of binding time is crucial to the design of an assembler. For each feature that you list as desirable, there is a price to be paid. Some desirable features will demand more storage during assembly, and storage may be in short supply; others would unduly lengthen the translation time.

The writer of the assembler must seek some sort of optimum that will benefit most users. The point is that an assembler cannot be viewed in isolation; it interacts with many other programs, some of which are out of the control of the programmer of the assembler. The items that interact are:

1. The source language.
2. The (relocatable) object language.
3. The link editor.
4. The loader.
5. The operating system.

Each of these influences the others, and there are trade-offs among them. For example, decisions in the assembly process can be delayed if the object language (and the link editor) are powerful enough. Thus, one can write a one-pass assembler and relegate binding of forward references to the link editor provided that one insists that all external references are declared in the program prelude (at the front). But this is a restriction on the user. It buys him fast assembly, but a complicated (but not necessarily slow) link editor (it will still be I/O limited). The appropriate approach is obviously in the eye of the beholder.

A second aspect is the size (and number of passes) of the assembler, the speed of assembly, and the size of the source program it can handle. As the amount of main storage used for tables is increased, the size of source program that can be assembled is decreased (without resorting to table overlay). Hence, as the instruction dictionary is expanded, or the number of characters per symbol is increased, or the number of macro skeletons required per program, or the verbosity of error messages (assuming that they are kept in main storage), the size of the program that can be assembled is reduced. As we progress down the line from mini to micro, this issue becomes more important.

The operating system is important in terms of the services it affords and restrictions it imposes on the assembler per se, as well as to the program being assembled. At a minimum, the assembler must provide linguistic elements (usually via a library of macros) that enable the programmer to call on the operating system services and specify exception handling (error and end conditions). Clearly, the syntax and semantics of these macros is dictated more by the operating system than by the writer of the assembler.

I take special issue with your remark that no assembler has ever satisfied your list of desirable features. Most modern assemblers do even more (with the possible exception of verbose error messages) functionally than you have asked for. Most modern assemblers include the following features:

1. The ability to specify your own mnemonic op-codes and synonyms for them, which is superior to any built-in synonym dictionary.

2. Symbols with simple constraints:

   a. Less than N characters (N varies from 6 on up).

   b. Restrictions on lead character (alpha or alphanumeric).

   c. Restrictions on embedded characters, usually no blanks or arithmetic operators (+, -, *, /).

3. Address arithmetic with, unfortunately, complex rules for expressions involving relocatable symbols--a point which you have ignored.

4. Detection of multiply defined, and undefined, symbols, with different error messages.

5. Plentiful pseudo-ops.

6. Conversion of constants.

7. Many list options.

8. Error messages, running the gamut from extremely concise to terribly verbose.

9. Listings with line numbers keyed to the compressed deck order and update (ALTER) corrections based on the line number. (Continuity numbers are meaningless without SOS-style CHANGE corrections, which are almost impossible to design in a human-engineered way.)

10. Commentary on instruction lines as well as independent commentary lines.

11. Built-in (operating system) macros; macros (as well as code) callable from a library, using INCLUDE.

12. User-defined in-line, or library-called macros.

13. Local symbols via various coding schemes (I would prefer different syntax and semantics in most cases).

14. Zero level addressing, including specification of decimal, floating, hex or octal, and character constants.

The equality

$$\left(\frac{9}{4}\right)^{\frac{27}{8}} = \left(\frac{27}{8}\right)^{\frac{9}{4}}$$

is pointed out in Problem 7-11 of Litton's Problematical Recreations (edited by James Hurley, 1971). What other numbers A and B have the property that

$$A^B = B^A ?$$

Z-Sequence
Z-Sequence
Z-Sequence

Dr. Mordecai Schwartz's work on Sequences (page 5 of this issue) triggered research into other possible ways of generating interesting sequences. Dr. Schwartz's sequences are intended to be predictable; that is, given enough terms, one can readily deduce the pattern of generation.

The twelve types of generator he used involve some mix of:

...the two previous terms
...the term number
...one or two constants

but there is plenty of room within that framework to seek new sequences. Some of these turn out to be dull, such as:

$$a_n = \left| a_{n-1} + a_{n-2} - n \right|$$

However, the sequence:

$$a_n = \left| 2a_{n-2} + a_{n-1} - n \right|$$

turns out to be fascinating. On the facing page are the first 156 terms of the sequence (taking the first two starting terms each to be unity). The sequence (call it Z) has these properties:

1	1	27	(14)	53	17	79	(64)	105	15	131	24
2	1	28	0	54	9	80	10	106	3	132	(106)
3	0	29	1	55	12	81	57	107	(74)	133	21
4	2	30	(29)	56	26	82	5	108	28	134	99
5	(3)	31	0	57	7	83	36	109	67	135	6
6	1	32	26	58	1	84	38	110	13	136	68
7	0	33	7	59	(44)	85	25	111	36	137	57
8	(6)	34	25	60	14	86	15	112	50	138	55
9	3	35	4	61	41	87	22	113	9	139	30
10	5	36	18	62	7	88	36	114	5	140	0
11	0	37	11	63	26	89	9	115	(92)	141	81
12	2	38	9	64	24	90	9	116	14	142	61
13	(11)	39	8	65	11	91	64	117	81	143	80
14	1	40	14	66	7	92	10	118	9	144	58
15	8	41	11	67	38	93	45	119	52	145	73
16	6	42	3	68	16	94	29	120	50	146	43
17	5	43	18	69	23	95	24	121	33	147	42
18	1	44	20	70	15	96	14	122	11	148	20
19	8	45	11	71	10	97	35	123	46	149	45
20	10	46	5	72	32	98	35	124	56	150	65
21	5	47	20	73	21	99	6	125	23	151	4
22	3	48	18	74	11	100	24	126	9	152	18
23	10	49	9	75	22	101	(65)	127	72	153	(127)
24	8	50	5	76	32	102	11	128	38	154	9
25	3	51	28	77	1	103	38	129	53	155	108
26	7	52	14	78	13	104	44	130	1	156	30

1. Successive terms seem unpredictable; that is, the values jump around, but not too wildly.

2. In the long run, its terms tend toward n/3, where n is the term number. Thus, the sum of the first 8002 terms is 11245744, whereas the sum of n/3 for n = 1, 2, 3,...,8002 is 11213501, a difference of only 32233.

3. The sequence is readily generated in any language on any machine, using the simplest of operations, and dealing only with positive integers. The accompanying table gives restart values at various points in the sequence.

4. No term after the first is ever as large as n. The circled numbers in the table show the appearances of new larger values. Such new larger values occur 68 times in the first 5081 terms.

5. Zero appears at terms 3, 7, 11, 28, 31, 140, 239, 600, and 6476, but does not seem to appear after that. Up to n = 5000, only 67 terms have a value less than 5.

6. Successive equal terms appear at n = 90 and n = 98, but this phenomenon does not seem to occur again.

7. If item 2 above holds indefinitely, then the sum of the series should be approximately the sum of n/3. The sum of each term divided by n should be approximately n/3 itself. The sum of each term divided by the square of the term number should be approximately 1/3 the sum of the harmonic series (see PC9-14). The sum of each term divided by the cube of the term number, however, should approximate 1/3 the sum of $(1/n^2)$, which should converge. It appears to converge to 1.2346...

8. Almost every integer appears as a term in the sequence sooner or later, except for a few numbers, such as 245, 449, 569, 575, and 903.

Various statistics that have already built up about the Z-sequence are given on the following pages. Items 5, 6, 7, and 8 above indicate the need for further exploration of this new function.

100	1530	2900	91450	5700	186112
200	4976	3000	92000	5800	182604
300	8286	3100	109134	5900	190098
400	12220	3200	111310	6000	202632
500	14702	3300	103512	6100	201228
600	18510	3400	113304	6200	213454
700	22532	3500	118204	6300	215678
800	24972	3600	114396	6400	205668
900	29744	3700	121572	6500	240600
1000	33124	3800	122420	6600	209158
1100	34840	3900	124572	6700	210156
1200	37634	4000	125240	6800	216194
1300	43490	4100	142410	6900	230826
1400	47800	4200	142714	7000	230776
1500	44824	4300	140472	7100	208950
1600	52124	4400	149084	7200	241294
1700	54240	4500	155144	7300	238912
1800	60852	4600	152844	7400	243070
1900	59192	4700	159518	7500	249480
2000	66428	4800	156766	7600	256620
2100	71988	4900	162708	7700	270472
2200	69482	5000	159268	7800	250886
2300	77658	5100	177284	7900	275186
2400	82708	5200	174080	8000	267358
2500	83586	5300	173936	8100	285022
2600	89790	5400	184918	8200	262086
2700	87008	5500	178608	8300	279974
2800	88490	5600	177888	8400	268484

Sums of successive groups of 100 terms of the Z-sequence.

Each entry can be compared to the corresponding sum of (n/3) by adding the hundred number to the preceding hundred number and multiplying by (50/3). For example, the last entry compares to (8400+8300)(50/3) = 275000. The discrepancy for the last entry is thus only 2.37%.

100	200	300	400	500	600	700	800	900
35	59	7	67	161	155	305	181	533
6	6	210	178	248	290	12	132	48
24	76	76	88	70	0	78	306	214

1000	1500	2000	2500	3000	3500	4000	4500
167	23	799	1615	1545	2149	1953	941
194	270	462	726	240	1028	1404	3524
472	1184	60	1456	330	1826	1310	906

5000	5500	6000	7000	8000	9000	10000	11000
2253	601	767	2517	7807	1291	4823	1655
340	2172	4654	3260	22	7422	3634	6818
154	2126	188	1294	7636	1004	3280	872

12000	13000	14000	15000
5753	4447	8309	6187
2080	3546	4884	2778
1586	560	7502	152

Table of restart values for the Z-sequence. The entry at 1500, for example, gives the values of the terms numbered 1498, 1499, and 1500.

Thus, 1500 - 2(23) - 270 = 1184.

The GILBREATH Conjecture

The accompanying figure illustrates the Gilbreath conjecture.* The left-hand column contains the consecutive prime numbers. Subsequent columns contain the absolute values of the differences. The leading number in each column of differences is one, and Norman Gilbreath conjectured that this phenomenon would hold good indefinitely. The article cited states that it has been verified for all primes less than 792 722 (that is, for the first 63 419 primes).

At first glance, it seems remarkable that the primes should behave so well, inasmuch as the primes distribute themselves irregularly among the natural numbers. What is really going on here begins with column 2, which has a leading 1 followed by small even numbers. For small even numbers, absolute differences will converge to 2 or 0 rapidly as one goes across a difference table. The whole pattern hinges on having the entries in column 2 relatively small in relation to what line they are on.

Consider the last row of the figure. How large would the left-hand entry have to get to disturb the 1 in the far right position? Sticking to the primes, it could get to 83 and still satisfy the pattern. That is to say, an entry as large as 22 on the 18th line of the first differences would be tolerable, as long as the entries above it were smaller. The first occurrence of a difference of 22 in the primes is at 1129.

a difference of		first appears with the prime shown here
	6	23
	10	139
	16	1831
	20	887
	30	4297
	40	19333
	50	31907
	100	396733
	150	13626257
	166	83751121
	200#	378043979
	220	47326693
	382#	10726904659

Source: "Statistics on the First Six Million Prime Numbers," F. Gruenberger and G. Armerding, October 1961, The RAND Corporation P-2460, for all entries except those marked (#), the latter due to T. R. Parkin and L. J. Lander, 1967.

*In "On a Conjecture Concerning the Primes," by R. B. Killgrove and K. E. Ralston in Mathematical Tables and Other Aids to Computation, 1959, p. 121.

In other words, the Gilbreath conjecture simply underscores the fact that differences between successive primes grow, but very slowly. The table shows some of the first appearances of differences. Thus, a difference of 100 will attenuate properly on the Gilbreath pattern, as long as there are about 100 columns of differences, which there would be at the line on which the prime 396733 would appear. So the conjecture is not quite so remarkable, but still interesting.

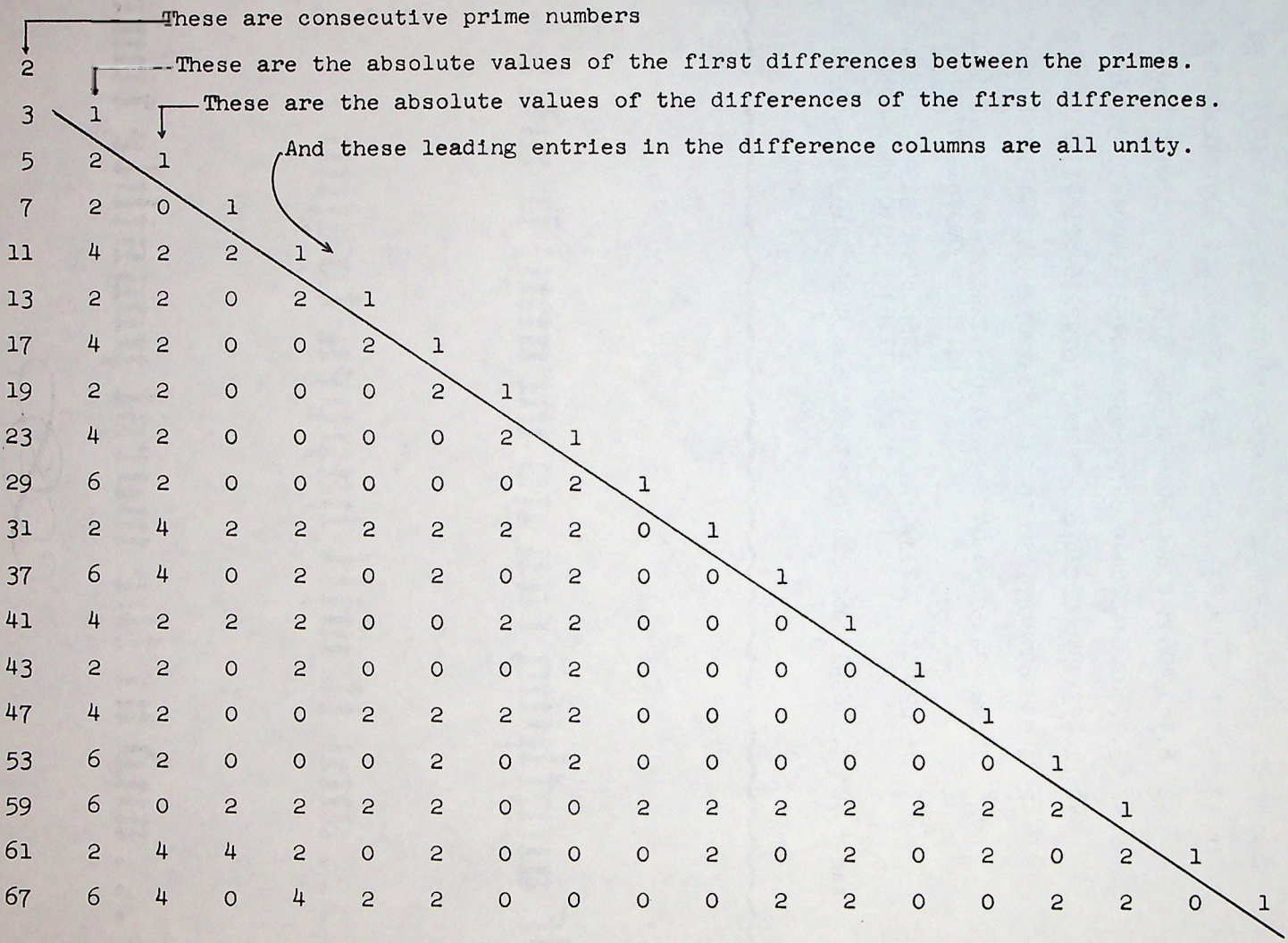
Which leads to a problem. Suppose we fabricate a new pattern as follows. Generate for the second column the sequence shown at the right:

and we extend this sequence to 1000 lines (the last entry will be 18, in the set beginning 2, 4, 6, 8, at line 992).

This pattern will satisfy the conjecture also; that is, if extended for absolute differences, the leading number for every column of differences will be <u>one</u>.

The problem is: how large can that last entry be before the pattern is spoiled?

This is exceptional:→ 1
2
2
4
2
4
6
2
4
6
8
2
4
6
8
10
2
4
6
8
10
12
.
.



2																
3	1															
5	2	1														
7	2	0	1													
11	4	2	2	1												
13	2	2	0	2	1											
17	4	2	0	0	2	1										
19	2	2	0	0	0	2	1									
23	4	2	0	0	0	0	2	1								
29	6	2	0	0	0	0	0	2	1							
31	2	4	2	2	2	2	2	2	0	1						
37	6	4	0	2	0	2	0	2	0	0	1					
41	4	2	2	2	0	0	2	2	0	0	0	1				
43	2	2	0	2	0	0	0	2	0	0	0	0	1			
47	4	2	0	0	2	2	2	2	0	0	0	0	0	1		
53	6	2	0	0	0	2	0	2	0	0	0	0	0	0	1	
59	6	0	2	2	2	2	0	0	2	2	2	2	2	2	2	1
61	2	4	4	2	0	2	0	0	0	2	0	2	0	2	0	2	1
67	6	4	0	4	2	2	0	0	0	0	2	2	0	0	2	2	0	1

PC42-20

Log 42	1.623249290397900463220983056572244529451891141976770
ln 42	3.737669618283368305917830101823882002360075421764866
$\sqrt{42}$	6.480740698407860230965967436087996657705204307058347
$\sqrt[3]{42}$	3.476026644886449786739865219004537434004838538786967
$\sqrt[10]{42}$	1.453198460282267816555796587579323698210413721730762
$\sqrt[100]{42}$	1.038083989471632675473129883720791394128425076802692
e^{42}	1739274941520501047.394681303611235226147984057725008 401037061011843950097817387725921
π^{42}	75909241720522939087 3.876361772680695986313489675624 4 891657911153243701079447449529 1
$\tan^{-1}$ 42	1.546991300609826675777453636281960677419304556603822

N-SERIES 42

If anything can go wrong, it will

...and it will happen to you

...and at the worst possible time